AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application.

1. (Currently Amended) A processor, comprising:

a circuit to receive a macro instruction specifying an operation, and specifying a first and a second data operand in first and second registers, respectively; and

one or more [[execution]] units to split the macro instruction into a first micro instruction and a second micro instruction, the first micro instruction specifying the operation on [[a first corresponding segment including]] a first portion of the first data operand and a first portion of the second data operand, and the second micro instruction specifying the operation on [[a second corresponding segment including]] a second portion of the first data operand and a second portion of the second data operand, and to execute the first micro instruction[[, and to execute]] at a different time than the second micro instruction.

2. (Original) The processor of claim 1, wherein the execution of the first micro instruction is performed during a first clock cycle.

3. (Currently Amended) The processor of claim [[1]] 2, wherein the execution of the second micro instruction is performed during a subsequent clock cycle.

4. (Original) The processor of claim 1, wherein the execution of the first micro instruction is performed during a first half clock cycle.

5. (Currently Amended) The processor of claim [[1]] 4, wherein the execution of the second micro instruction is performed during a second half clock cycle.

6. (Currently Amended) The processor of claim 1, further comprises a register file coupled with the circuit, the register file [[having]] to store a third data operand including a plurality of data elements corresponding to the first data operand and the second data operand.

7. (Currently Amended) The processor of claim 6, wherein [[the plurality of data elements]] each of the operands includes 128-bits of data from the register file.

Claims 8 - 10 (Cancelled)

11. (Currently Amended) A method, comprising:

receiving a macro instruction specifying an operation, and specifying a first and a second data operand in first and second registers, respectively; and

splitting the macro instruction into a first micro instruction and a second micro instruction, the first micro instruction specifying the operation on [[a first corresponding segment including]] a first portion of the first data operand and a first portion of the second data operand, and the second micro instruction specifying the operation on [[a second corresponding segment including]] a second portion of the first data operand and a second portion of the second data operand.

12. (Original) The method of claim 11, further comprising:

executing the first micro instruction; and

executing the second micro instruction.

13. (Original) The method of claim 12, wherein the execution of the first micro instruction is performed during a first clock cycle, and the execution of the second micro instruction is performed during a second clock cycle.

Claim 14 (Cancelled)

15. (Previously Presented) A method, comprising:

receiving a macro instruction specifying an operation, and specifying first and second data operands in first and second registers, respectively;

performing the operation on lower order data segments of the first and second data operands on a first set of execution units; and

performing the operation on high order data segments of the first and second data operands on the first set of execution units.

16. (Currently Amended) The method of claim 15, further comprises [[splitting]] dividing the macro instruction into a first micro instruction and a second micro instruction.

17. (Currently Amended) The method of claim 16, wherein the first micro instruction [[specifying]] specifies the operation on [[a first corresponding segment including]] a first portion of the first data operand and a first portion of the second data operand.

18. (Currently Amended) The method of claim 16, wherein the second micro instruction [[specifying]] specifies the operation on [[a second corresponding segment including]] a second portion of the first data operand and a second portion of the second data operand.

19. (Previously Presented) The method of claim 16, further comprising:

executing the first micro instruction; and

executing the second micro instruction.

Claims 20-24 (Cancelled)

25. (Currently Amended) A processor, comprising:

a circuit to receive a macro instruction specifying an operation, and specifying first and second data operands in first and second registers, respectively; and

an execution unit in communication with the circuit, the execution unit to perform the operation on lower order data segments of the first and second data operands on a first set of execution units, and to perform the operation on high order data segments of the first and second data operands on the first set of execution units.

26. (Currently Amended) The processor of claim 25, [[wherein the execution unit is further to split]] further comprising a decoder to divide the macro instruction into a first micro instruction and a second micro instruction.

27. (Currently Amended) The processor of claim 26, wherein the first micro instruction [[specifying]] specifies the operation on a first corresponding segment including a first portion of the first data operand and a first portion of the second data operand.

28. (Currently Amended) The processor of claim 26, wherein the second micro instruction [[specifying]] specifies the operation on a second corresponding segment including a second portion of the first data operand and a second portion of the second data operand.

29. (Previously Presented) The processor of claim 26, wherein the execution unit is further to:

execute the first micro instruction; and

execute the second micro instruction.

Claims 30-44 (Cancelled)

45. (New) A method comprising:

receiving a single macro instruction specifying at least two logical registers, wherein the two logical registers respectively store first and second 128-bit packed data operands, each of the first and second 128-bit packed data operands have four 32-bit single precision floating point data elements; and

independently performing an operation specified by the single macro instruction on a first and a second plurality of corresponding ones of the 32-bit single precision floating point data elements of the first and second 128-bit packed data operands, at different times, using the same circuit, to independently generate a first and a second plurality of resulting data elements, wherein the first and the second plurality of resulting data elements are stored in a single logical register as a third 128-bit packed data operand.

46. (New) The method of claim 45, wherein said independently performing comprises:

accessing full widths of the first and second 128-bit packed data operands;

splitting the full widths of each of the first and second 128-bit packed data operands into lower and higher halves that each include two 32-bit single precision floating point data elements; and

delaying the higher halves of each of the first and second 128-bit packed data operands.

47. (New) The method of claim 45, wherein said independently performing comprises:

converting the single macro instruction into a first micro instruction and a second micro instruction;

accessing lower halves of the first and second 128-bit packed data operands and performing an operation specified by the first micro instruction on each of two corresponding pairs of the 32-bit single precision floating point data elements of the lower halves; and

accessing higher halves of the first and second 128-bit packed data operands and performing the operation specified by the second micro instruction on each of two corresponding pairs of the 32-bit single precision floating point data elements of the higher halves.

48. (New) An apparatus comprising:

a register file to contain first and second 128-bit packed data operands, the first and the second 128-bit packed data operands including four pairs of corresponding 32-bit single precision floating point data elements; and

a circuit coupled to the register file, the circuit in response to a single packed data instruction specifying an operation to:

retrieve the corresponding data elements from the register file;

execute the operation in an execution unit on a lower order two of the four pairs of corresponding 32-bit single precision floating point data elements to output a first result including two 32-bit single precision floating point data elements;

at a different time, execute the operation in the execution unit on a higher order two of the four pairs of corresponding 32-bit single precision floating point data elements to output a second result including two 32-bit single precision floating point data elements; and

store the first and the second results in the register file as a third 128-bit packed data operand.

49. (New) The apparatus of claim 48, further comprising:

ports of the circuit to receive full widths of the first and the second 128-bit packed data operands;

logic to divide the full widths of each of the first and the second 128-bit packed data operands into the lower and the higher order two; and

delay elements coupled with the logic to delay the higher order two.

50. (New) The apparatus of claim 48, further comprising a decoder to convert the single packed data instruction into a first micro instruction that causes the lower order two to be accessed from the register file and a second micro instruction that causes the higher order two to be accessed from the register file.

51. (New) A method comprising:

receiving a first packed data instruction, the first packed data instruction specifying logical registers in a 128-bit logical register file of a processor storing a first 128-bit packed data operand and a second 128-bit packed data operand, the first packed data instruction also specifying an operation to be performed on corresponding 32-bit single precision floating point data elements of the first and the second 128-bit packed data operands, each of the 128-bit packed data operands including a lower order half and a higher order half, each of the lower order half and the higher order half including two of the 32-bit single precision floating point data elements; and

performing the operation specified by the first packed data instruction on the corresponding 32-bit single precision floating point data elements of the lower order halves of the first and the second 128-bit packed data operands using a circuit; and

at a different time, performing the operation specified by the first packed data instruction on the corresponding 32-bit single precision floating point data elements of the higher order halves of the first and the second 128-bit packed data operands using the circuit.

52. (New) The method of claim 51, further comprising:

receiving a second packed data instruction, the second packed data instruction specifying logical registers in the 128-bit logical register file of the processor storing a third 128-bit packed data operand and a fourth 128-bit packed data operand, the second packed data instruction also specifying an operation to be performed on corresponding 64-bit data elements of the third and the fourth 128-

bit packed data operands, each of the third and the fourth 128-bit packed data operands including a lower order half and a higher order half, each of the lower order half and the higher order half of the third and the fourth 128-bit packed data operands including one of the 64-bit data elements; and

performing the operation specified by the second packed data instruction on the corresponding 64-bit data elements of the lower order halves of the third and the fourth 128-bit packed data operands using the circuit; and

at a different time, performing the operation specified by the second packed data instruction on the corresponding 64-bit data elements of the higher order halves of the third and the fourth 128-bit packed data operands using the circuit.

53. (New) The method of claim 51:

wherein the circuit comprises a 64-bit execution unit; and

wherein the operation is one of ADD and MULTIPLY.

54. (New) The method of claim 51, further comprising:

converting the first packed data instruction into at least a first micro instruction and a second micro instruction;

accessing the lower order halves of the first and the second 128-bit packed data operands from the logical registers responsive to the first micro instruction; and

accessing the higher order halves of the first and the second 128-bit packed data operands from the logical registers responsive to the second micro instruction.

55. (New) The method of claim 51, further comprising, prior to said performing the operation, accessing full widths of the first and the second 128-bit packed data operands from the logical registers.

56. (New) The method of claim 51, wherein the operation of the first packed data instruction comprises an ADD operation, and further comprising scheduling a second packed data instruction, which specifies a MULTIPLY operation, out-of-order so that it follows the first packed data instruction.

57. (New) An apparatus comprising:

a plurality of physical registers to operate as a 128-bit logical register file of a processor;

a decoder to receive and decode instructions including a packed data instruction that specifies a first 128-bit packed data operand and a second 128-bit packed data operand by specifying logical registers in the 128-bit logical register file, each of the 128-bit packed data operands including a lower order half and a higher order half, each of the lower order half and the higher order half including two 32-bit single precision floating point data elements, the packed data instruction also specifying an operation to be performed on corresponding ones of the 32-bit single precision floating point data elements of the first and the second 128-bit packed data operands; and

an execution unit, coupled with the decoder, to execute the packed data instruction to generate a 128-bit packed data result operand by performance of the operation specified by the packed data instruction on the corresponding ones of the 32-bit single precision floating point data elements in the lower order halves of the first and the second 128-bit packed data operands, and, at a different time,

performance of the operation specified by the packed data instruction on the corresponding ones of the 32-bit single precision floating point data elements of the higher order halves of the first and the second 128-bit packed data operands.

58. (New) The apparatus of claim 57:

wherein the decoder is to receive and decode a second packed data instruction that specifies a third 128-bit packed data operand and a fourth 128-bit packed data operand by specifying logical registers in the 128-bit logical register file, each of the third and the fourth 128-bit packed data operands including a lower order half and a higher order half, each of the lower order half and the higher order half of the third and the fourth 128-bit packed data operands including a 64-bit data element, the second packed data instruction also specifying an operation to be performed on corresponding ones of the 64-bit data elements of the third and the fourth 128-bit packed data operands; and

wherein the execution unit is to execute the second packed data instruction to generate a second 128-bit packed data result operand by performance of the operation specified by the second packed data instruction on the corresponding ones of the 64-bit single precision floating point data elements in the lower order halves of the third and fourth 128-bit packed data operands, and, at a different time, performance of the operation specified by the second packed data instruction on the corresponding ones of the 64-bit single precision floating point data elements of the higher order halves of the third and the fourth 128-bit packed data operands.

59. (New) The apparatus of claim 57:

wherein the execution unit comprises a 64-bit execution unit;

wherein the operation is one of ADD and MULTIPLY; and

wherein the 128-bit packed data result operand is to be stored over the first 128-bit packed data operand.

60. (New) The apparatus of claim 57, wherein the decoder is to convert the packed data instruction into at least a first micro instruction and a second micro instruction, wherein the first micro instruction causes the lower order halves of the first and the second 128-bit packed data operands to be accessed from the logical registers, and wherein the second micro instruction causes the higher order halves of the first and the second 128-bit packed data operands to be accessed from the logical registers.

61. (New) The apparatus of claim 57, wherein the execution unit comprises logic to access full widths of the first and the second 128-bit packed data operands from the logical registers, and wherein the execution unit comprises delay elements to delay the higher order halves of the first and the second 128-bit packed data operands while performing the operation on the lower order halves of the first and the second 128-bit packed data operands.

62. (New) The apparatus of claim 57, further comprising a scheduling unit, coupled between the decoder and the execution unit, to try to schedule packed data instructions that specify ADD and MULTIPLY operations on 128-bit single precision floating point operands out-of-order so that they alternate.

63. (New) A method comprising:

receiving a single macro instruction that specifies an operation to be independently performed on corresponding 32-bit single precision floating point

data elements from a first 128-bit packed data operand and a second 128-bit packed data operand to generate a third 128-bit packed data operand;

performing the operations on lower halves of the first and second 128-bit packed data operands at a different time than on upper halves of the first and second 128-bit packed data operands using the same hardware; and

storing results of the four operations as four 32-bit single precision floating point packed data elements of the third 128-bit packed data operand.

64. (New) A method comprising:

receiving a packed data instruction specifying logical registers storing a first 128-bit packed data operand and a second 128-bit packed data operand;

accessing full widths of the first 128-bit packed data operand and the second 128-bit packed data operand from the logical registers;

splitting the full width of the first 128-bit packed data operand into a first 64-bit lower order segment and a first 64-bit higher order segment, the first 64-bit lower order segment and the first 64-bit higher order segment each having two 32-bit single precision floating point data elements;

splitting the full width of the second 128-bit packed data operand into a second 64-bit lower order segment and a second 64-bit higher order segment, the second 64-bit lower order segment and the second 64-bit higher order segment each having two 32-bit single precision floating point data elements;

generating a first result by using a circuit to perform an operation specified by the packed data instruction on corresponding ones of the 32-bit single precision

floating point data elements of the first 64-bit lower order segment and the second 64-bit lower order segment; and

at a different time, generating a second result by using the circuit to perform the operation specified by the packed data instruction on corresponding ones of the 32-bit single precision floating point data elements of the first 64-bit higher order segment and the second 64-bit higher order segment.

65. (New) An apparatus comprising:

a first port to receive a full width of a first 128-bit packed data operand from a logical register specified by a packed data instruction, the first 128-bit packed data operand having four 32-bit single precision floating point data elements;

a second port to receive a full width of a second 128-bit packed data operand from a logical register specified by the packed data instruction, the second 128-bit packed data operand also having four 32-bit single precision floating point data elements;

a first circuit coupled with the first port to receive the full width of the first 128-bit packed data operand and to split the first 128-bit packed data operand into a first 64-bit lower order half and a first 64-bit higher order half;

a second circuit coupled with the second port to receive the full width of the second 128-bit packed data operand and to split the second 128-bit packed data operand into a second 64-bit lower order half and a second 64-bit higher order half;

a first delay element coupled with the first circuit to receive and delay the first 64-bit higher order half;

a second delay element coupled with the second circuit to receive and delay the second 64-bit higher order half;

an execution unit coupled with the first circuit, the second circuit, the first delay element, and the second delay element, the execution unit to perform an operation specified by the packed data instruction on corresponding ones of the 32-bit single precision floating point data elements of the first and the second 64-bit lower order halves, and, after a delay, to perform the operation specified by the packed data instruction on corresponding ones of the 32-bit single precision floating point data elements of the first and the second 64-bit higher order halves.

66. (New) A method comprising:

receiving a packed data instruction specifying logical registers having a first 128-bit packed data operand having four 32-bit single precision floating point data elements and a second 128-bit packed data operand having four 32-bit single precision floating point data elements;

converting the packed data instruction into a first micro instruction and a second micro instruction;

retrieving lower order halves of the first and the second 128-bit packed data operands specified by the first micro instruction, each of the lower order halves having two 32-bit single precision floating point data elements;

generating a first result by using hardware to perform an operation specified by the packed data instruction on corresponding ones of the 32-bit single precision floating point data elements of the lower order halves of the first and the second 128-bit packed data operands;

retrieving higher order halves of the first and the second 128-bit packed data operands specified by the second micro instruction, each of the higher order halves having two 32-bit single precision floating point data elements;

generating a second result by using the hardware to perform an operation specified by the packed data instruction on corresponding ones of the 32-bit single precision floating point data elements of the higher order halves of the first and the second 128-bit packed data operands;

writing the first and the second results to a 128-bit logical register specified by the packed data instruction.

67. (New) An apparatus comprising:

a decoder to receive a packed data instruction, the packed data instruction specifying logical registers having a first 128-bit packed data operand and a second 128-bit packed data operand, each of the first and the second 128-bit packed data operands having four 32-bit single precision floating point data elements, the decoder to convert the packed data instruction into at least a first micro instruction and a second micro instruction;

an execution unit coupled with the decoder to execute the first micro instruction, and, at a different time, to execute the second micro instruction,

in the execution of the first micro instruction the execution unit to retrieve lower order halves of the first and the second 128-bit packed data operands and to perform an operation specified by the packed data instruction on corresponding ones of the 32-bit single precision floating point data elements in the lower order halves, and

in the execution of the second micro instruction the execution unit to retrieve higher order halves of the first and the second 128-bit packed data operands and to perform the operation specified by the packed data instruction on corresponding ones of the 32-bit single precision floating point data elements in the higher order halves.

68. (New) A method comprising:

receiving a first packed data instruction specifying logical registers that respectively store a first 128-bit packed data operand and a second 128-bit packed data operand, each of the first and the second 128-bit packed data operands having four 32-bit single precision floating point data elements, the first packed data instruction specifying an ADD operation on the first and the second 128-bit packed data operands;

receiving a second packed data instruction specifying logical registers that respectively store a third 128-bit packed data operand and a fourth 128-bit packed data operand, each of the third and the fourth 128-bit packed data operands having four 32-bit single precision floating point data elements, the second packed data instruction specifying a MULTIPLY operation on the third and the fourth 128-bit packed data operands;

at a first time, performing the ADD operation specified by the first packed data instruction on corresponding ones of the 32-bit single precision floating point data elements in lower order halves of the first 128-bit packed data operand and the second 128-bit packed data operand using an ADD execution unit;

at a second time, which is different than the first time, performing the ADD operation specified by the first packed data instruction on corresponding ones of

the 32-bit single precision floating point data elements in higher order halves of the first 128-bit packed data operand and the second 128-bit packed data operand using the ADD execution unit;

at the second time, performing the MULTIPLY operation specified by the second packed data instruction on corresponding ones of the 32-bit single precision floating point data elements in lower order halves of the third 128-bit packed data operand and the fourth 128-bit packed data operand using a MULTIPLY execution unit; and

at a third time, which is different than the second time, performing the MULTIPLY operation specified by the second packed data instruction on corresponding ones of the 32-bit single precision floating point data elements in higher order halves of the third 128-bit packed data operand and the fourth 128-bit packed data operand using the MULTIPLY execution unit.

69. (New) The method of claim 68, further comprising executing code having instructions with operations arranged in the pattern ADD, MULTIPLY, ADD, MULTIPLY.

70. (New) The method of claim 68, further comprising scheduling packed data instructions out-of-order so that packed data instructions specifying MULTIPLY operations are interleaved with packed data instructions specifying ADD operations.